Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 383 (2007) 5–22
Theoretical Computer Science
www.elsevier.com/locate/tcs

Centralized asynchronous broadcast in radio networks☆

Bogdan S. Chlebus[a,*], Mariusz A. Rokicki[b]

[a] *Department of Computer Science and Engineering, University of Colorado at Denver and Health Sciences Center, Denver, CO 80217, USA*
[b] *Centre National de la Recherche Scientifique, LRI UMR 8623, Université Paris Sud, 91405 Orsay Cedex, France*

Abstract

We study asynchronous broadcasting in packet radio networks. A radio network is represented by a directed graph, in which one distinguished source node stores a message that needs to be disseminated among all the remaining nodes. An asynchronous execution of a protocol is a sequence of events, each consisting of simultaneous deliveries of messages. The correctness of protocols is considered for specific adversarial models defined by restrictions on events the adversary may schedule. A protocol specifies how many times the source message is to be retransmitted by each node. The total number of transmissions over all the nodes is called the work of the broadcast protocol; it is used as complexity measure. We study computational problems, to be solved by deterministic centralized algorithms, either to find a broadcast protocol or to verify the correctness of a protocol, for a given network. The amount of work necessary to make a protocol correct may have to be exponential in the size of network. There is a polynomial-time algorithm to find a broadcast protocol for a given network. We show that certain problems about broadcasting protocols for given networks are complete in NP and co-NP complexity classes.

Keywords: Radio network; Asynchrony; Broadcast protocol; Centralized algorithm; Lower bound; Complexity class; Completeness

1. Introduction

Broadcast is among the most natural and useful communication primitives. One node of a network is designated as a source while the other nodes are assumed to be reachable from the source along directed paths. The source holds a message that needs to be disseminated among all the nodes in the network. Such a dissemination is achieved by relaying the message from node to node.

We consider asynchronous broadcasting in packet radio networks. Radio networks are represented by directed graphs. An asynchronous execution of a broadcast protocol is a sequence of events, each representing deliveries of messages that occur simultaneously. The correctness of a protocol at hand is understood in the context of a given adversary who schedules events with the goal to prevent a successful completion of broadcast.

☆ A preliminary version of this paper appeared as "Asynchronous broadcast in radio networks", in: Proceedings, 11th Colloquium on Structural Information and Communication Complexity (SIROCCO), in: Lecture Notes in Computer Science, vol. 3104, Springer-Verlag, Heidelberg, 2004, pp. 57–68. This work was supported by the NSF Grant 0310503.

* Corresponding author.
E-mail address: Bogdan.Chlebus@cudenver.edu (B.S. Chlebus).

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.03.057

A broadcast protocol for a given radio network specifies, separately for each node, how many times the source message is to be retransmitted by the node. The total number of transmissions over all the nodes is called the work of the broadcast protocol and is used as a complexity measure. We study computational problems, related to asynchronous radio broadcasting, to be solved by deterministic centralized algorithms. One problem is about finding a broadcast protocol and another is to verify the correctness of a protocol, for a given network.

Models of radio networks and synchrony. Radio networks studied in the literature have been assumed to be synchronous. Synchrony in networks means an environment in which time is partitioned into rounds, while nodes are controlled by local clocks ticking at the same rate. A stronger assumption specifies that local clocks produce the same round numbers. Every transmission by a node of a synchronous radio network occurs within one round and occupies the whole round. A message sent at a round is delivered to the recipients in the same round. A message delivered to a node may sometimes not be *heard*, that is received successfully, in a radio network. The motivation for this is that all messages are transmitted on radio waves of the same frequency and so multiple messages arriving at the same round to a node interfere with one another. The key properties of the synchronous mode of radio communication can be summarized as follows:

(i) If a message is sent by a node at a round, then the message is delivered at this round to *all* the out-neighbors of the node.
(ii) A message arriving at a round to a node is heard by the node when it is the *only* message that arrived to this node at this round.

This paper proposes how to extend the synchronous mode of radio communication to capture asynchrony. The assumption of synchrony may appear to be essential for radio networks: the model is defined by the property that messages arriving to a node at the same round collide and none of them can be received successfully. Observe that the possibility of a collision of messages requires only the ability to categorize transmissions as either occurring simultaneously or not. In a synchronous scenario, both notions 'simultaneous' and 'in the same round' mean the same. We abstract from round numbers and work with the notion 'to occur simultaneously'. An execution of a protocol is defined to consist of a sequence of abstract events. Any such event represents what occurs simultaneously. This approach is related to the methodology applied when considering asynchrony and synchronization in theory of distributed computing; see the books on this topic by Attiya and Welch [2], Lynch [21], and Taubenfeld [22].

We propose to restrict asynchrony by way of adversaries who control the timing of arrivals of messages. We study three specific adversaries, called edge, crash and node ones. The *edge* adversary is the most powerful one in that timings of deliveries of messages may be independent for each traversed *edge*. The *crash* adversary is defined to be the edge adversary augmented with the power to *crash* some of the nodes before the start of an execution. The *node* adversary is restricted by the property that all deliveries of copies of a message generated by a single transmission of a *node* occur simultaneously.

Motivation. The standard approach to study radio networks is valid in environments satisfying a number of tacit assumptions. One of them is that the nodes given in the network's specification are the only ones participating in communication protocols. Another assumption is that communication protocols can control the behavior of the nodes to the extent that a protocol may be specified as a schedule that determines for each node and each round whether a transmission by the node occurs or not at the round. Our motivation in introducing asynchrony is to capture communication scenarios when these assumptions are relaxed.

In one of the proposed models, asynchrony occurs because edges do not represent transmission ranges, as typically is the case in the literature, but rather disjoint paths in a larger network. The graph specified as the network under consideration is actually a subnetwork of a larger communication structure, and consists of only the nodes that are responsible for achieving the communication goal. The communicating units omitted from the network's specification play the role of auxiliary relay stations. These stations are busy performing other tasks, but join from time to time to help route traffic by forwarding messages. The following are natural assumptions about relay stations. Every message received by a relay station is eventually retransmitted. Transmissions by relay stations can interfere with other transmissions only at the nodes of the explicitly considered network. A relay station forwards each received message exactly once. If one wants a relay station to repeat a transmission a certain number of times, then the way to make it happen is to have the relay station receive the message the same number of times. These properties provide a

scenario in which a message transmitted by a node of a radio network will arrive at each of its neighbors with some unpredictable delay, which may depend on the neighbor. This model is formally defined by way of the edge adversary, see Section 2.

In the second model, all nodes are explicitly listed in the network's specification, but the nodes execute communication protocols concurrently with performing other tasks. When a message arrives at a node, it is stored temporarily, and then sent at some future round. Nodes are not expected to follow a protocol so that an execution for a node is a sequence of transmissions at rounds known in advance. However, when a node does perform a transmission, then this results in all its out-neighbors receiving the message simultaneously. This model is formally defined by way of the node adversary, see Section 2.

The third model incorporates failures. It is obtained by augmenting an adversary with the power to crash nodes. We consider only the case when the edge adversary can crash nodes before the start of an execution of a protocol. This model is formally defined by way of the crash adversary, see Section 2.

Summary of results. We propose how to study asynchrony in radio networks. The main ingredients of our approach include (1) abstracting an execution as a sequence of simultaneous transmissions called events, and (2) imposing restrictions on events in the same execution to capture additional properties of the model. Possible executions of a broadcast protocol are defined by way of adversarial models. We consider three specific adversaries, called edge, node and crash ones. We show how to find, for a given input network, an asynchronous broadcast protocol for the network. The work complexity of the obtained protocol, measured as the total number of transmissions, may be exponential. We show next that this is an inherent property of the model by proving exponential lower bounds for each of the considered adversaries. The problem to verify if a given protocol is correct, against either the edge or the crash adversary, is shown to be polynomially solvable. The problem to decide if there is a protocol of at most a given work complexity which is correct against the edge adversary is shown to be NP-complete. A notable difference between the edge and node adversaries is that while the correctness against the edge adversary, for a given network and protocol, is polynomially solvable, this problem for the node adversary is shown to be complete in co-NP.

Previous work. Broadcasting problems in radio networks have been considered for a number of settings. On one hand, a network could be given as an input to a centralized algorithm, whose task is to produce a broadcast protocol tailored to the network. On the other hand, the only part of specification of a network could be its size, so that a protocol is expected to be correct for all networks of a given size. The studied protocols could be either deterministic or randomized. Graphs representing networks have been assumed to be either directed or undirected. The only common assumption that has always been made seems to be the synchronous mode of operation. In this overview D denotes the maximum length of a shortest path from the source, Δ the maximum in-degree, and n the size, which is the number of nodes. A radio network is directed unless stated otherwise. A protocol is *explicit* if it can be produced in time that is polynomial in n, otherwise it is called *existential*.

The model of radio communication was introduced by Chlamtac and Kutten [4]. They showed NP-completeness of finding an optimal broadcasting schedule for a given radio network. A lower bound $\Omega(\log^2 n)$ for broadcasting in radio networks was given by Alon, Bar-Noy, Linial and Peleg [1], it holds even for networks of a bounded diameter. A lower bound $\Omega(n \log D)$ was shown by Clementi, Monti and Silvestri [10].

Chlamtac and Weinstein [5] developed a deterministic sequential algorithm that finds in polynomial time a broadcast protocol working in $\mathcal{O}(D \log^2 n)$ time, for a given network. Clementi, Crescenzi, Monti, Penna and Silvestri [9] developed a polynomial-time algorithm producing a protocol working in $\mathcal{O}(D \log \Delta \log(n/D))$ time for a given network. A centralized polynomial-time algorithm can produce a protocol working in $\mathcal{O}(D + \text{polylog } n)$ time for a given undirected network, as was shown by Gaber and Mansour [12] and Gąsieniec, Peleg and Xin [14].

The round-robin distributed protocol completes broadcasting in $\mathcal{O}(n^2)$ time. The first distributed deterministic explicit broadcast protocol with sub-quadratic time performance was given by Chlebus, Gąsieniec, Gibbons, Pelc and Rytter [7]. Their protocol was first improved to time performance $\mathcal{O}(n^{3/2})$ by Chlebus, Gąsieniec, Östlin and Robson [8], and next by Indyk [15] to $\mathcal{O}(n^{1+o(1)})$. The fastest known distributed deterministic broadcast of time performance $\mathcal{O}(n \log^2 D)$ was given by Czumaj and Rytter [11]; this protocol is not explicit.

The first distributed randomized broadcast protocol of sub-quadratic expected-time performance were given by Bar-Yehuda, Goldreich and Itai [3]. The fastest known randomized protocols working in the expected $\mathcal{O}(D \log(n/D) + \log^2 n)$ time were developed by Czumaj and Rytter [11] and by Kowalski and Pelc [16]. Kushilevitz and Mansour [19]

showed a lower bound $\Omega(D\log(n/D))$ on the expected time of broadcasting. This fact combined with the lower bound $\Omega(\log^2 n)$ shows that $D\log(n/D)+\log^2 n$ is the asymptotically optimum time complexity of a randomized broadcast.

Deterministic distributed broadcasting in undirected networks was studied by Kowalski and Pelc [16] who developed a protocol of time performance $\mathcal{O}(n\log n)$ and showed a lower bound $\Omega(n\log n/\log(n/D))$. Fault-tolerant broadcasting in radio networks was studied by Kushilevitz and Mansour [20] and by Kranakis, Krizanc and Pelc [18]. The impact on efficiency of distributed radio broadcast of information that may be a part of code and of randomness in protocols has been extensively studied, see [3,16,17].

Structure of this document. The paper is organized as follows. Section 2 discusses the technical notions, in particular asynchronous executions. We discuss ways to obtain a broadcast protocol, for a given radio network, in Section 3. Section 4 includes exponential lower bounds on the work of broadcast protocols, with correctness understood with respect to any of the considered adversaries. In Section 5 we develop polynomial-time algorithms to verify if a given broadcast protocol is correct, for the edge and crash adversaries, respectively. Section 6 contains a discussion of problems regarding asynchronous radio broadcast that are complete in NP and co-NP complexity classes, respectively. We conclude with final remarks in Section 7.

2. Preliminaries

A radio network is modeled as a directed graph $G=(V,E)$, where $V=V[G]$ is a set of nodes, and $E=E[G]$ is a set of directed edges. An edge $\langle x,y\rangle\in E$ is also denoted as $x\to y$. When $x\to y$ is an edge, then node x is its *start point,* and node y is its *end point*, while x is an *in-neighbor* of y and y an *out-neighbor* of x. The set of all the in-neighbors of y is denoted as $\mathrm{IN}(y)$, and the set of all out-neighbors of x is denoted as $\mathrm{OUT}(x)$. We assume that a network is represented by lists of pointers to in- and out-neighbors of each node.

For a node x and each outgoing edge $x\to y$, there is the *outgoing buffer* $\mathrm{out}_x(y)$ at x associated with the edge. When node x *sends* a message, then a number of copies of the message is put into each such a buffer $\mathrm{out}_x(z)$ at x, for an outgoing edge $x\to z$. For a node y and an incoming edge $x\to y$, there is the *incoming buffer* $\mathrm{in}_y(x)$ at y associated with the edge. When a message is removed from the outgoing buffer $\mathrm{out}_x(y)$ and placed in the incoming buffer $\mathrm{in}_y(x)$, then the message is said to be *delivered from x to y*.

Broadcasting occurs as a sequence of events. An *event* consists of a number of deliveries and sendings of the source message along edges of the underlying network. Each node sends in at most one event in an execution, but this may result in multiple events of delivery to the same out-neighbor. The deliveries in an event are interpreted as occurring simultaneously. The number of deliveries in an event is always positive, except for the initial event when the source performs the first transmission. At most one message can traverse any specific edge from an out-going to the in-coming buffer in an event.

We distinguish, for a transmitted message, to be 'delivered' from being 'successfully received'. A message delivered to a node x in an event is *heard* at x if it is the only message delivered to x in this event. If more than one messages arrive at a node in an event, then none is received successfully by the node. It is possible that some nodes hear a message in an event while others do not. Our definition of events allows nodes to send the source message just after it was heard for the first time. The following is a legitimate scenario: the source node is heard at a node x in a certain event and x sends the message in the same event.

A broadcast protocol for graph G is determined by a function $\mathcal{B}:V[G]\to\mathbb{N}$, which assigns a non-negative integer $\mathcal{B}(v)$ to each node $v\in V[G]$, with the additional requirement that $\mathcal{B}(s)>0$ for the source s. Such a function is called *repeat-broadcast function* and the value $\mathcal{B}(v)$ is called *the repeat-broadcast value of v*. The protocol given by such a function determines how nodes send messages: when a node v sends a message, then this results in placing $\mathcal{B}(v)$ copies of the message into outgoing buffers of all the outgoing edges. We often refer to the broadcast protocol determined by a repeat-broadcast function $\mathcal{B}$ by the same name $\mathcal{B}$.

Events are components of executions of protocols. An *execution of protocol $\mathcal{B}$* is defined to be a sequence $\mathcal{E}=\langle e_0,e_1,\ldots,e_\ell\rangle$ of events, which satisfies the following properties:

1. The first event e_0 of $\mathcal{E}$ consists of sending the source message by the source s.
2. All the outgoing buffers are empty after the last event e_ℓ.

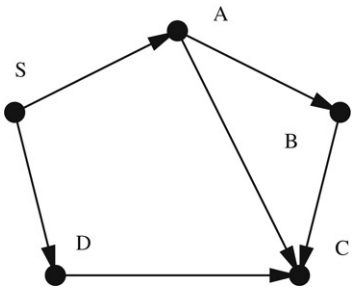

Fig. 1. A radio network with source S. The repeat-broadcast function equal to 1 at each node gives a protocol that is correct against the node adversary but not against the edge one.

If a node v with repeat-broadcast value $\mathcal{B}(v)$ hears the source message at some event, then deliveries of the message from v to any out-neighbor have to occur in $\mathcal{B}(v)$ events. If, on the other hand, v does not hear the source message at all in an execution, then a delivery of the messages from v to its out-neighbors does not occur at any event in this execution.

An execution is *fair* in the sense that any copy of the source message that is placed in an outgoing buffer is eventually delivered. An execution terminates only when there is nothing to be delivered. An execution is said to be *successful* when every node in the network has heard the source message.

Further restrictions on executions are given in the framework of adversarial models. The motivation is to capture additional properties of an asynchronous mode of radio communication. The correctness of protocols is always understood for specific adversaries. Adversaries are convenient but not necessary, since we could simply restrict the class of executions in the same way as the adversary is defined. We consider three specific adversarial models.

Edge adversary: Acts subject only to the general definition of an execution of a given broadcast protocol.

Crash adversary: It is the edge adversary augmented with the power to fail nodes by crashing. Crashes occur prior to the start of an execution of an algorithm. The faulty nodes do not participate in executions, and may be considered removed from the graph. This may make some nodes unreachable: such nodes are also considered crashed.

Node adversary: Acts subject to the following additional constraint: if the source message is received at a node from some node x, then the message is received at all the out-neighbors of x at the same event.

When a message from node v is received in an event in an execution controlled by the node adversary, then we say that v *transmitted* in this event. If a protocol with $\mathcal{B}(v) = k$ is executed and the node v receives the message, then v transmits exactly k times in the execution.

Given a broadcast protocol $\mathcal{B}$ and an adversary $\mathcal{A}$, protocol $\mathcal{B}$ is *correct against* $\mathcal{A}$ when every execution of $\mathcal{B}$ consistent with the power of $\mathcal{A}$ is successful. The node adversary is more restricted than the edge one. A protocol that is correct against the node adversary may not be correct against the edge one. As an example, consider the graph in Fig. 1 and the protocol with the repeat-broadcast function equal to 1 at each node. To see why this protocol is correct against the node adversary, consider two cases. If the message from the nodes A and D is delivered to C at the same event, then node B hears the message at the same event, but a delivery from B to C cannot be blocked by a delivery from D. Otherwise the first delivery of the message to C is heard by C. This protocol is not correct against the edge adversary though, because it is possible that the message is delivered to C from all A, B and D in the same event.

The *work* $\mathcal{W}(\mathcal{B})$ of a protocol $\mathcal{B}$ is defined to be equal to the sum of the repeat-broadcast values of all nodes of G, that is, $\mathcal{W}(\mathcal{B}) = \sum_{v\in V} \mathcal{B}(v)$. Observe that the length of an execution is at most $\sum_{v\in V} \mathcal{B}(v) \deg^+(v)$ in the case of the edge adversary, where $\deg^+(v)$ is the out-degree of v, since just one message may be delivered in each event. This length is at most $\mathcal{W}(\mathcal{B}) = \sum_{v\in V} \mathcal{B}(v)$ in the case of the node adversary.

The definition of asynchrony we assume allows for any instance of a radio network with a distinguished source to have a correct broadcast protocol, as is demonstrated in Section 3. This is not the case for gossiping. A problem of *gossiping* has each node start with its *rumor* and the goal is for every node to learn all the rumors. Conceptually, gossiping is equivalent to multiple concurrent instances of broadcasting.

Proposition 1. *There is a strongly connected radio network for which no repeat-broadcast function determines a gossiping protocol correct against the node adversary.*

Proof. Consider a network of three nodes a, b and c, in which any ordered pair of nodes in each of the sets $\{a, b\}$ and $\{b, c\}$ is connected by a directed edge. In particular, nodes a and c are connected to b but there is no edge between a and c. The only way for the node a to learn the rumor of the node c, is to hear it relayed by node b. Similarly, the only way for the node c to learn the rumor of the node a, is to hear it relayed by node b. The node b might not learn all the rumors in some execution. To see this, consider two cases for a protocol $\mathcal{B}$. If $\mathcal{B}(a) = \mathcal{B}(c)$, then each delivery from a to b can be matched by a delivery from c to b, and so b may not learn any of the remaining rumors. Otherwise, when $\mathcal{B}(a) \neq \mathcal{B}(c)$, assume without loss of generality that $\mathcal{B}(a) < \mathcal{B}(c)$. Then any delivery from a to b can be matched by a delivery from c to b, so b may not learn the rumor of a. □

3. Protocol by enforcing dominance

A node x is called a *relay for node* y if $x \to y$ is an edge and the distance from the source to x is smaller than to y. Let $S \subseteq V$ be a set of nodes. A node $x \in S$ is said to *dominate* S if the inequality

$$\mathcal{B}(x) > \sum_{y \in S \setminus \{x\}} \mathcal{B}(y)$$

holds, where the empty sum is assumed to be equal to zero. Each node different from the source has at least one relay node, because all nodes are reachable. If node x is an in-neighbor of y and x dominates all in-neighbors of y, then x is said to *guard* the node y, which is also denoted by $x = L(y)$. A node x is *guarded* if some in-neighbor of x guards x.

Lemma 1. *If, for each node y different from the source, there is a relay node that guards y, then the protocol is correct with respect to any among the considered adversaries.*

Proof. We show, by induction on the distance from the source to y, that y eventually obtains the message. If the distance is zero, then $y = s$ and so y holds the message. Suppose the distance is positive, let x be a relay node that guards y. By the inductive assumption, x receives the message eventually. At least one transmission from x cannot be matched and blocked by a transmission from another neighbor of y. This holds because otherwise the total number of messages at the remaining in-neighbors of y would be at least $\mathcal{B}(x)$. □

Lemma 1 cannot be strengthened by assuming only that each node different from the source is guarded by some neighbor. There are also protocols correct against the edge adversary in which some nodes are not guarded by a neighbor.

Next we describe an algorithm DISTANCE-DOMINATION to find a protocol correct against the edge adversary. The algorithm assigns values of repeat-broadcast function to nodes of a network $G = (V, E)$ given as input.

Let us fix an enumeration $\langle v_i \rangle_{1 \leq i \leq n}$ of all the nodes of G such that if the distance from the source s to v_k is smaller than the distance from the source to v_ℓ then $\ell < k$. In particular, we have that $s = v_n$. Initialize the broadcast function $\mathcal{B}(v) = 0$, for each node $v \in V$. The function is next modified for each node, in the increasing order of indices. Take an index i, such that $1 \leq i \leq n$. Suppose all $1 \leq k < i$ have already been processed. Define

$$C_j = 1 + \sum_{x \in \mathrm{IN}(v_j)} \mathcal{B}(x)\,. \tag{1}$$

We set $\mathcal{B}(v_i)$ equal to the maximum value among C_j over all edges $v_i \to v_j$, if at least one such an edge exists, and $\mathcal{B}(v_i) = 0$ otherwise.

A pseudocode of algorithm DISTANCE-DOMINATION is given in Fig. 2. The summation uses a different range of indices than the one in (1), because $\mathcal{B}(v_i)$ plays the role of a local variable to store the largest value of $z = C_j$ examined so far.

Theorem 1. *Given a graph with $n = |V|$ nodes and $m = |E|$ edges, the algorithm* DISTANCE-DOMINATION *works in $\mathcal{O}(n+m)$ time and produces a protocol that is correct against the edge adversary. The work of the obtained protocol is always smaller than 2^{n-1}.*

Proof. A tree of shortest paths from the source can be found in $\mathcal{O}(n + m)$ time by breadth-first search. Nodes can be ordered by their distances from the source in $\mathcal{O}(n)$ time by traversing the tree. The main loop of the algorithm takes time $\mathcal{O}(n + m)$, because the time is proportional to the sum of the adjacency lists.

```
Algorithm DISTANCE-DOMINATION ( G, s )
    for each node v ∈ V[G] do B(v) := 0
    find a tree of shortest paths from s
    sort the nodes in reversed order on the distance from the source s
    let ⟨v_1, ..., v_n = s⟩ be the resulting ordering
    for i := 1 to n do
        for each v_j ∈ OUT(v_i) do
            z := 0
            if j < i then z := 1 + Σ_{x∈IN(v_j)\{v_i}} B(x)
            if B(v_i) < z then B(v_i) := z
```

Fig. 2. Algorithm DISTANCE-DOMINATION which finds a repeat-broadcast function $\mathcal{B}$ for a given network G with source s.

We show, by reversed induction on the distance from the source $v_n = s$ to a node v_j, that there is a relay node for v_j that guards v_j. Let v_i be a relay of v_j of the largest index among such relay nodes. Then $i > j$ and, by the definition of the repeat-broadcast function, the value of $\mathcal{B}(v_i)$ equal to

$$\mathcal{B}(v_i) = 1 + \sum_{x \in \mathrm{IN}(v_j) \setminus \{v_i\}} \mathcal{B}(x)$$

was assigned last among values assigned to all the in-neighbors of v_j. It follows that the node v_i dominates all the in-neighbors of v_j. This shows that Lemma 1 is applicable, which yields correctness.

Next we show by induction on the indices of the nodes in the numbering $\langle v_i \rangle_{1 \le i \le n}$ that the inequality $\mathcal{B}(v_i) \le 2^{i-1}$ holds. For $i = 1$, we have $\mathcal{B}(v_1) = 0 \le 2^0$, because v_1 is not a relay for any node. Consider $i > 1$ and let $v_i \to v_j$ be an edge. At the time when $\mathcal{B}(v_i)$ is computed, the values $\mathcal{B}(x) > 0$ among $\mathrm{IN}(v_j)$ occur only at $x = v_k$ for $k < i$. Hence we may use the inductive assumption $\mathcal{B}(v_k) \le 2^{k-1}$, and estimate

$$\begin{aligned}
\mathcal{B}(v_i) &= 1 + \sum_{x \in \mathrm{IN}(v_j)} \mathcal{B}(x) \\
&\le 1 + \sum_{1 \le k < i} \mathcal{B}(v_k) \\
&\le 1 + \sum_{1 \le k < i} 2^{k-1} \\
&= 1 + 2^{i-1} - 1 \\
&= 2^{i-1} .
\end{aligned}$$

This shows that $\mathcal{B}(v_i) \le 2^{i-1}$ for all v_i. The work $\sum_{1 \le i \le n} \mathcal{B}(v_i)$ of the protocol can be upper-bounded by the sum $\sum_{1 \le i \le n} 2^{i-1} = 2^n - 1$, which is smaller than 2^n. □

The work performance of the protocol produced by the algorithm DISTANCE-DOMINATION is exponential, which cannot be improved in general, as we show in Section 4. The obtained protocol is also *a fortiori* correct against the node adversary.

One could observe that first sorting the nodes by the distance to them from the source in increasing order of indices, and then assigning 2^i to a node v_i as a value of a repeat-broadcast function, results in a correct protocol. This is because any set of nodes has a dominating element, hence Lemma 1 applies. If this is so simple, why to develop the algorithm DISTANCE-DOMINATION? The answer is that for some natural classes of graphs the work of a protocol obtained by this algorithm is $o(2^n)$, for networks of n nodes.

Theorem 2. *For networks with in-degrees at most d, the work of the protocol obtained by the algorithm* DISTANCE-DOMINATION *is $\mathcal{O}(C^n)$, for some C such that $1 < C < 2(1 - 2^{-d})$.*

Proof. Follow the proof of Theorem 1. Numbers $\mathcal{B}(v_i)$ satisfy the inequality

$$\mathcal{B}(v_{m+1}) \leq \sum_{k=m-d}^{m} \mathcal{B}(v_k) ,$$

for $i \geq d$. Hence the inequality $\mathcal{B}(v_i) \leq A_i$ holds, where the numbers A_i are defined by the recurrence relation

$$A_{k+d+1} = A_{k+1} + A_{k+2} + \cdots + A_{k+d}$$

and suitable initial values. This is a definition of the d-round Fibonacci numbers. Its characteristic equation $x^d - x^{d-1} - x^{d-2} - \cdots - x - 1 = 0$ has one positive simple root C in the interval $[1, 2]$ that satisfies the inequality $1 < C < 2(1 - 2^{-d})$, see [23]. □

Algorithm DISTANCE-DOMINATION is not applicable against the crash adversary. In this case, assigning 2^i to a node v_i as a value of a repeat-broadcast function results in a correct protocol. This approach is best possible as we show in Section 4.2 by proving a matching lower bound.

4. Lower bounds

In this section we show that there are networks for which any broadcast protocol requires exponential work.

4.1. Edge and node adversaries

We construct a family of networks G_n, each with n nodes, such that, for any algorithm on G_n correct against the edge adversary, its work has to be at least $c^{n^{1/3}}$, for some constant $c > 1$.

We first define a family of networks F_k, for each integer $k > 0$. The network F_k has $\binom{k}{3} + k + 1$ nodes, partitioned into three layers. The *top layer* L_0 consists of only the source node. The *middle layer* L_1 consists of k nodes. The *bottom layer* L_2 includes the remaining $\binom{k}{3}$ nodes. Directed edges connect the source with each element of layer L_1. There is a one-to-one correspondence between three-element subsets of L_1 and the nodes of the layer L_2. For each three-element subset X of L_1, there is exactly one node v in the layer L_2 such that X consists of in-neighbors of v.

Lemma 2. *Let F_k be the network as defined above, for $k > 0$. If a repeat-broadcast function for the nodes of F_k determines a broadcast algorithm correct against the edge adversary, then, for each node v in the bottom layer, there is a guard for v in the middle layer.*

Proof. Let $\mathcal{B}$ be the repeat-broadcast function. Assume, by way of a contradiction, that there is no guard for a node v. Let v_1, v_2 and v_3 be the in-neighbors of v, where the inequalities $\mathcal{B}(v_1) \geq \mathcal{B}(v_2) \geq \mathcal{B}(v_3)$ hold. The adversary may use the following strategy to block v. First deliver the message to all the in-neighbors of v and then have each transmission of v_1 collide with a transmission by some other node. This is achievable because v_1 is not a guard for v. Namely, the first transmissions of v_1 can be blocked just by v_2, and node v_3 joins at the first round when there are $\mathcal{B}(v_3) \geq \mathcal{B}(v_1) - \mathcal{B}(v_2)$ messages still to be sent by v_1, and continues till the end. This shows that the repeat-broadcast function does not define a correct algorithm, which is a contradiction. □

For a broadcast protocol for the network F_k, we can assume that it is consistent with indexing of the nodes in the middle layer, in the sense that $\mathcal{B}(v_i) \leq \mathcal{B}(v_j)$ if and only if $i \leq j$, for $1 \leq i, j \leq k$, because otherwise we could renumber the nodes.

Lemma 3. *Let $\mathcal{B}$ be a broadcast protocol for F_k correct against the edge adversary. Then the following inequalities hold: $\mathcal{B}(v_1) \geq 0$ and $\mathcal{B}(v_i) \geq f_{i-2}$, for $k \geq i \geq 2$, where f_i is the i-th Fibonacci number.*

Proof. Induction on $k \geq 3$. For $k = 3$, the middle layer consists of just three nodes $L_1 = \{v_1, v_2, v_3\}$ and the bottom layer consists of one node $L_2 = \{u\}$, while each node from L_1 is an in-neighbor of u. The least expensive among correct broadcast protocols is defined as follows: $\mathcal{B}(s) = \mathcal{B}(v_3) = 1$ and $\mathcal{B}(v_1) = \mathcal{B}(v_2) = 0$. A base of induction holds, because $f_0 = 0$ and $f_1 = 1$.

Assume that the claim holds for F_k. Consider a repeat-broadcast function $\mathcal{B}$ that works for F_{k+1}. The subgraph induced by the nodes $s, v_1, \ldots, v_k$ and the nodes that correspond to all three-element subsets of $\{v_1, \ldots, v_k\}$ is

isomorphic to F_k. By the inductive assumption, we have that $\mathcal{B}(v_1) \geq 0$ and $\mathcal{B}(v_i) \geq f_{i-2}$, for $2 \leq i \leq k$. By Lemma 2, there is a guard in $\{v_{k-1}, v_k, v_{k+1}\}$ for the corresponding node in the bottom layer. By the numbering of the nodes, which makes the repeat-broadcast function monotonous, the only possible candidate for a guard is the node v_{k+1}. We obtain

$$\begin{aligned}\mathcal{B}(v_{k+1}) &\geq \mathcal{B}(v_k) + \mathcal{B}(v_{k-1}) + 1\\ &\geq f_{k-2} + f_{k-3} + 1\\ &\geq f_{k-1}\ ,\end{aligned}$$

which completes the proof. □

Theorem 3. *There is a sequence $\langle G_n\rangle_{n\geq 1}$ of networks, such that G_n has n nodes and any broadcast protocol for this network that is correct against the edge adversary requires $\Omega(\phi^{\sqrt[3]{n}})$ work, where $\phi = (1+\sqrt{5})/2$ is the golden ratio.*

Proof. Take $k = \lfloor n^{1/3} \rfloor$ and consider F_k. The network F_k is of size $\Theta(n)$. By Lemma 3 and properties of Fibonacci numbers, the work of the algorithm is at least

$$\sum_{i=1}^{k} \mathcal{B}(v_i) \geq \sum_{i=2}^{k} f_{i-2} = f_k - 1\ .$$

Since f_k is equal to $\phi^k/\sqrt{5}$ rounded to the nearest integer, the work of the algorithm is at least $\Omega(\phi^{\sqrt[3]{n}})$. □

The node adversary is weaker than the edge one, in the sense that each algorithm correct against the edge adversary is also correct against the node one.

Corollary 1. *There is a sequence $\langle G_n\rangle_{n\geq 1}$ of networks, each G_n of n nodes, such that any broadcast protocol for G_n that is correct against the node adversary requires work $\Omega(c^{\sqrt[3]{n}})$, for a constant $c > 1$.*

Proof. Modify the graphs used in a proof of Theorem 3 by inserting an additional node in the middle of each edge. The power of the node adversary on such graphs is equal to that of the edge one. □

4.2. Crash adversary

We show that $\Omega(2^n)$ of work is sometimes necessary to accrue for a protocol that is correct against the crash adversary. This amount of work is then optimal, since it is always achievable, as observed in Section 3.

Let $\mathcal{B}$ be a broadcast protocol and v be a node in the network distinct from the source. The node v is said to be *blockable for* $\mathcal{B}$ if the crash adversary can crash some nodes different from the source, possibly none, so that node v is reachable from the source in the obtained network but does not receive the source message in a certain execution of $\mathcal{B}$. A protocol $\mathcal{B}$ is correct for the given network if there is no node blockable for $\mathcal{B}$ in the network.

Lemma 4. *Let $\mathcal{B}$ be a broadcast protocol for network G. If, for each node, any set of its in-neighbors contains a guard, then the protocol $\mathcal{B}$ is correct for G against the crash adversary.*

Proof. Let us assume that some node v of G is blockable for $\mathcal{B}$. Let the adversary crash some nodes in such a way that v is reachable but v does not receive the message in some execution. Take a shortest path $(s, u_1, \ldots, u_{k-1}, u_k = v)$ from the source s to v. Let $i \leq k$ be the smallest number such that u_i does not receive the message in some execution of the protocol on this network. The set of all non-faulty in-neighbors of u_i that eventually receive the message in this execution contains a guard. This implies that the message gets eventually delivered to u_i, which is a contradiction. □

We say that node v is *vulnerable* if a removal of an arbitrary proper subset $S \subset \text{IN}(v)$ of nodes of v and also of the node v does not disconnect the nodes in $\text{IN}(v) \setminus S$ from the source s. Intuitively, a node v is vulnerable if, in a situation when any proper subset of neighbors of v has been crashed, the remaining in-neighbors of v can receive the message before v receives it.

Lemma 5. *Let a node v of G be vulnerable. If protocol $\mathcal{B}$ is correct for G, then each subset of* $\text{IN}(v)$ *contains a guard.*

Proof. Let us assume that protocol $\mathcal{B}$ is correct and the node v has no guard in some proper subset $S \subset \text{IN}(v)$. Then the adversary has the following strategy to block v: first, it fails the nodes in $\text{IN}(v) \setminus S$; next it delivers the message to all the nodes in S but not to v. This can be done because node v is vulnerable and protocol $\mathcal{B}$ is correct. Since there is no guard in S, all transmissions from the nodes in S might collide. Hence v is blockable, which is a contradiction. □

Lemma 6. *Let $V = \{v_0, \ldots, v_{n-1}\}$ and $\mathcal{F} : V \to \mathbb{N}$, where $\mathcal{F}(v_0) \leq \mathcal{F}(v_1) \leq \cdots \leq \mathcal{F}(v_n)$. If $\mathcal{F}$ has the property that for each subset $S \subseteq V$ there exists an element $v \in S$ such that v dominates S, then the inequality $\mathcal{F}(v_i) \geq 2^i$ holds.*

Proof. For $n = 1$, the set V contains just one node v_0 and $\mathcal{F}(v_0)$ has to be at least $1 = 2^0$. This provides the base of induction. Let us assume that what we are to prove is true for $n - 1$, and consider n. The inequalities $F(v_i) \geq 2^i$, for $0 \leq i \leq n-1$, hold by the inductive assumption. There is a dominating node in $\{v_0, \ldots, v_n\}$, by the assumption. Since $\mathcal{F}(v_n) \geq F(v_i)$, for $0 \leq i \leq n-1$, we have that v_n is the only node that can dominate $\{v_0, \ldots, v_n\}$. The smallest possible value for $\mathcal{F}(v_n)$ is $1 + \sum_{i=0}^{n-1} 2^i = 2^n$. □

Theorem 4. *There exists a network G of n nodes for which any protocol $\mathcal{B}$ correct against the crash adversary attains $\Omega(2^n)$ work.*

Proof. Consider a network H defined as follows. It consists of n nodes partitioned into three layers. The top layer L_0 consists of one source node s. The middle layer L_1 consists of $n - 2$ nodes. The bottom layer L_2 consists of one node v. The node s is the in-neighbor of each node from layer L_1 and each node from L_1 is the in neighbor of the node v.

The node v is vulnerable, so by Lemma 5 there exists a guard in each subset of L_1. By Lemma 6, the smallest possible amount of work is achieved when the nodes in L_1 transmit $2^0, 2^1, \ldots, 2^{n-3}$ times, respectively. Hence the amount of work is $\Omega(2^n)$. □

5. Verifying correctness of protocols

Given a network G with a source s and a repeat-broadcast function $\mathcal{B}$, we consider the problem how to verify if the broadcast protocol determined by $\mathcal{B}$ is correct against the given adversary.

5.1. Correctness against the edge adversary

For two nodes u and v, we say that *node u depends on v* if any path from the source s to u traverses either v or a node w with $\mathcal{B}(w) = 0$. If an edge $u \to v$ has the property that u depends on v, then such an edge is said to be *redundant*. An edge that is not redundant is called *significant*. If the edge $u \to v$ is redundant, then transmissions by u cannot block v from receiving the source message, since node v receives the message before u. Redundant edges can be removed without affecting the set of nodes that receive the source message, for any behavior of the adversary.

We may start with removing all redundant edges. To identify redundant edges, do the following. For each node v, first remove v and all nodes w with $\mathcal{B}(w) = 0$, and then identify all nodes than cannot be reached from the source by a directed path. If node u is among them and $u \to v$ is an edge of G then this edge is redundant. In the remaining part of this section we assume that all edges are significant.

Lemma 7. *If node v does not have a guard, then the adversary can block v from receiving the message.*

Proof. Let z be an in-neighbor of v with a maximum value $\mathcal{B}(z)$ among all the in-neighbors. The adversary has the message delivered to each in-neighbor of v but not to v. This can be done because there are no redundant edges. Next the adversary enforces each message sent by z to collide with at least one message sent from some other in-neighbor of v. This is achievable because z is not a guard. □

Lemma 7 implies that every node has to have a guard for the protocol to be correct. This is a necessary but not a sufficient condition however, see Fig. 3.

Lemma 8. *If there is a guard of node v, and the guard receives the message, then v also will receive the message.*

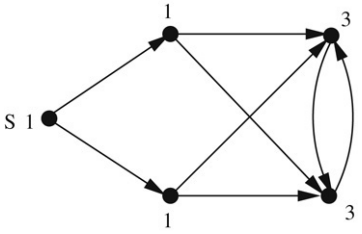

Fig. 3. A broadcast protocol in which each node has a guard while the two rightmost nodes can be blocked. Numbers by the nodes denote repeat-broadcast values.

```
procedure CHECK-NODE-BLOCKED ( v )
    if v = s then return false
    BlockedNodes := {v} ; B(v) := 0
    remove all redundant edges
    repeat
        NewNodesToBlock := ∅
        for each node u ∈ BlockedNodes
            if L(u) exists then
                NewNodesToBlock := NewNodesToBlock ∪ {L(u)}
                B(L(u)) := 0
                remove all redundant edges
        BlockedNodes := BlockedNodes ∪ NewNodesToBlock
    until NewNodesToBlock = ∅
    if s ∈ BlockedNodes then return true else return false
```

Fig. 4. A procedure to verify if node v can be blocked by the edge adversary.

Proof. At least one transmission by $L(v)$ cannot be blocked by a neighbor of v, because the node $L(v)$ dominates them all. □

Now we develop a routine to verify if a node v can be blocked from receiving a message. A pseudocode for this routine is in Fig. 4, where $L(v)$ denotes a function that returns a guard of node v. Guards may change as the repeat-broadcast function is evolving.

The routine maintains a set BlockedNodes, which stands for the set of *blocked nodes*. The name is a shorthand for a description of a set of nodes that cannot receive the message if the node v is to be blocked. Initialize BlockedNodes $= \{v\}$ and modify the function $\mathcal{B}$ so that $\mathcal{B}(v) = 0$ since v will never broadcast as a blocked node. Remove all redundant edges for this modified $\mathcal{B}$. Lemma 8 implies that $L(v)$ should not receive a message if v is to be blocked. Add the node $L(v)$ to the set BlockedNodes and modify the repeat-broadcast function $\mathcal{B}$ so that $\mathcal{B}(L(v)) = 0$. Now remove all redundant edges for such a modified $\mathcal{B}$. Next, for each node x in BlockedNodes, find its guard $L(x)$ and add the guard to the BlockedNodes while modifying $\mathcal{B}$ to be equal to zero on $L(x)$. Keep iterating this until at some point the set BlockedNodes stops to increase, which occurs when there are no more guards to be blocked. If the source s is in BlockedNodes, then the node v cannot be blocked, otherwise it can.

Lemma 9. *The procedure* CHECK-NODE-BLOCKED *is correct.*

Proof. We define two invariants for the two nested loops: the *repeat-until-invariant* of the outer repeat loop, and the *for-invariant* of the inner for loop.

REPEAT-UNTIL-INVARIANT:
The set BlockedNodes contains the nodes that cannot receive the message if the node v is to be blocked. The set NewNodesToBlock contains all nodes that cannot receive a message, in order to block all the nodes in BlockedNodes $\setminus$ NewNodesToBlock. All redundant edges, defined by a repeat-broadcast function which assigns zero to all nodes in BlockedNodes, have been removed.

```
Algorithm VERIFY-EDGE-CORRECT ( G, B )
    Correct := true
    for each node v ∈ V[G] do
        if CHECK-NODE-BLOCKED(v) then Correct := false
    if Correct then return "given repeat-broadcast function is correct"
        else return "given repeat-broadcast function is not correct"
```

Fig. 5. Algorithm VERIFY-EDGE-CORRECT to verify correctness of repeat-broadcast function $\mathcal{B}$, for the underlying network G, against the edge adversary.

We prove this invariant by induction on the number of iterations of the outer loop. The base is by the following two cases.

Case 1: There is no guard for the node v.

Then BlockedNodes $= \{v\}$ and NewNodesToBlock $= \emptyset$ and $\mathcal{B}(v) = 0$ and all redundant edges are removed.

Case 2: There is a guard $L(v)$ for node v.

Then BlockedNodes $= \{v, L(v)\}$ and NewNodesToBlock $= \{L(v)\}$ and $\mathcal{B}(v) = \mathcal{B}(L(v)) = 0$ and all redundant edges are removed.

The repeat-until-invariant holds in both of these cases. Let us assume that we enter the repeat loop and the repeat-until-invariant holds.

> FOR-INVARIANT:
> Before the ith iteration the following properties hold:
> The set BlockedNodes $= \{v_1, v_2, \ldots, v_{i-1}, v_i, \ldots, v_k\}$ contains nodes that cannot receive the message if the node v is to be blocked. The set NewNodesToBlock contains the nodes that cannot receive the message if the nodes v_1 through v_{i-1} in BlockedNodes are to be blocked. All redundant edges, defined with the assumption that the nodes in BlockedNodes ∪ NewNodesToBlock do not transmit, have been removed.

When the for-loop is completed, then the for-invariant holds and the set BlockedNodes equals BlockedNodes ∪ NewNodesToBlock, so the repeat-until-invariant holds too. After we exit the repeat loop, the set BlockedNodes contains all nodes that cannot receive the message, in order to block v, since the set NewNodesToBlock is empty. If BlockedNodes contains the source s then the adversary has no strategy to block v, since the source always performs at least one transmission. If BlockedNodes does not contain s, then the adversary can make the nodes in BlockedNodes never receive the message, so v is blocked. □

Algorithm Verify-Edge-Correct given in Fig. 5 verifies the correctness of a protocol by resorting to procedure Check-Node-Blocked.

Theorem 5. *Given a network G of n nodes, and a repeat-broadcast function $\mathcal{B}$, the algorithm* VERIFY-EDGE-CORRECT *checks if the protocol $\mathcal{B}$ is correct against the edge adversary in $\mathcal{O}(n^6)$ time.*

Proof. Each node v is added to the set BlockedNodes at most once, since after v has been put in BlockedNodes, its repeat-broadcast value is set to zero and v is never a guard. The repeat loop can be executed up to n times. The inner for loop takes |BlockedNodes| iterations. Each iteration consist of finding $L(u)$, which takes $\mathcal{O}(|\text{IN}(u)|)$ time per call, followed by removing the redundant edges in $\mathcal{O}(n^3)$ time. The runtime of CHECK-NODE-BLOCKED is $\mathcal{O}((1 + 2 + \cdots + n)n^3) = \mathcal{O}(n^5)$, which yields the total $\mathcal{O}(n^6)$ time. □

5.2. Correctness against the crash adversary

We want to be able to verify if a protocol $\mathcal{B}$ for a given network G is correct.

Lemma 10. *If there exists a node that is blockable by the crash adversary, then there exist a node u that is blockable by crashing only some neighbors of u.*

```
Algorithm VERIFY-AGAINST-CRASHES ( B )
    remove all redundant edges
    for each node v do
        C := B
        while L(v) exists for C do
            C(L(v)) := 0
            remove redundant edges for C
        if IN(v) is nonempty then
            return "v is blockable"
    return "the protocol is correct"
```

Fig. 6. Algorithm VERIFY-AGAINST-CRASHES to verify correctness of $\mathcal{B}$ against the crash adversary.

Proof. Suppose that node v is blockable. There exists a path $(s = u_0, u_1, \ldots, u_k = v)$ of non-faulty nodes and a node u_j such that all nodes in $(u_j, \ldots, u_k = v)$ do not receive the message, while all nodes in $(s = u_0, \ldots, u_{j-1})$ do receive it. Let $S \subseteq \text{IN}(u_j)$ be the set of non-faulty neighbors of u_j. Let $W \subset S$ be a subset of non-faulty in-neighbors that are blocked. The nodes in W do not transmit hence there is no guard in $S \setminus W$ to block u_j. The adversary has the following strategy to block u_j by crashing only its in-neighbors: the nodes in $S \setminus W$ are not crashed and the remaining in-neighbors of u_j in $\text{IN}(u_j)$ \ $(S \setminus W)$ are crashed.

The adversary has the message delivered to all the nodes in $S \setminus W$. Next all transmissions from the nodes in $S \setminus W$ are scheduled to collide, which is achievable due to the absence of a guard in $S \setminus W$. This means that u_j is blocked as required. □

Next we consider an algorithm to verify the correctness of a protocol, by way of checking if there exists a blockable node. For each node v check if there is a subset $S \subseteq \text{IN}(v)$ such that S does not contain a guard. If for some node v such a subset S is found, then this v is blockable, and the protocol is not correct, otherwise $\mathcal{B}$ is correct. The algorithm is in Fig. 6.

Theorem 6. *The algorithm* VERIFY-AGAINST-CRASHES*($\mathcal{B}$) checks in $\mathcal{O}(n^5)$ time if a protocol $\mathcal{B}$ for a network of n nodes is correct.*

Proof. The correctness follows from Lemma 10. The outer for-loop takes n iterations. The inner while-loop takes at most n iterations. Removing redundant edges costs $\mathcal{O}(n^3)$ time. The total runtime is thus $\mathcal{O}(n^5)$. □

6. Computationally hard problems

We present apparently infeasible decision problems regarding broadcasting in asynchronous networks. The problems are hard in complexity classes located at the lowest levels of the polynomial hierarchy [13].

The lowest levels of the polynomial hierarchy are defined as follows. Class P consists of problems solvable in polynomial time. Class $\Sigma_1^{\text{P}} = \text{NP}$ consists of problems such that an instance has a certificate verifiable in polynomial time. Class $\Pi_1^{\text{P}} = \text{co-NP}$ consists of languages whose complements are in $\Sigma_1^{\text{P}} = \text{NP}$. Class $\Sigma_2^{\text{P}} = \text{NP}^{\text{NP}}$ consists of problems for which instances have certificates such that the verification of their validity is a problem in $\Pi_1^{\text{P}} = \text{co-NP}$.

6.1. Hard for the edge adversary

We will resort to NP-completeness of the following problem, see [13].

Problem: EXACT COVER BY 3-SETS (3XC)
Instance: A set $S = \{x_1, \ldots, x_{3m}\}$ and a family of sets $F = \{C_1, \ldots, C_k\}$, where each C_i is a subset of S of size three.
Question: Can S be covered by all sets in some $H \subseteq F$ in such a way that each $x_i \in S$ belongs to exactly one $C_j \in H$?

The first problem is about existence of a broadcast protocol with work bounded locally.

Problem: EDGE-ADVERSARY 0–1 RADIO BROADCAST
Instance: A network G with a distinguished source.
Question: Is there a broadcast protocol for G correct against the edge adversary and such that each node transmits at most once?

Theorem 7. *The problem* EDGE-ADVERSARY 0–1 RADIO BROADCAST *is* NP*-complete.*

Proof. The problem is in NP, because it is sufficient to guess a protocol of a 0–1 transmission pattern and apply the algorithm developed in Section 5 to verify that it is correct.

Next we give a reduction of 3XC to the problem considered. Take an instance of 3XC an build a network of three layers as follows. The top layer L_0 contains just the source s. The middle layer L_1 contains nodes V_i, for each C_i. The bottom layer L_2 contains nodes Y_j, for each x_j. The source s is an in-neighbor of each node in L_1. The node V_i of the middle layer is an in-neighbor of Y_j if and only if $x_j \in C_i$. There is a one-to-one correspondence between correct 0–1 broadcasting algorithms and covering sub-families $H \subseteq F$. Namely, we declare a set C_i to be in H if and only if the repeat-broadcast function on V_i equals 1. The property of S being covered by all sets in H is translated into a property that each node in the bottom layer will have the message delivered from at least one node in the middle layer. The property of disjointness of the elements of H translates into a property that each node in the bottom layer can have a message delivered from at most one neighbor in the middle layer, which guarantees that the message will be heard. □

The next problem is about existence of a broadcast protocol with work bounded globally.

Problem: EDGE-ADVERSARY WORK-BOUNDED RADIO BROADCAST
Instance: A network G with a distinguished source, and a positive integer W.
Question: Is there a broadcast protocol for G correct against the edge adversary and such that its work is at most W?

Theorem 8. *The problem* EDGE-ADVERSARY WORK-BOUNDED RADIO BROADCAST *is* NP*-complete.*

Proof. The problem is in NP, because it is sufficient to guess a protocol of a total work at most W and apply the algorithm developed in Section 5 to confirm that it is correct. Next we give a reduction of 3XC to the problem considered. Take an instance of 3XC and build a network of three layers as in the proof of Theorem 7. Set the parameter W equal to m, which is one third of the size of S. A broadcast protocol correct for this network and of work W is determined by a repeat-broadcast function that assigns only values of 0 and 1. □

6.2. Hard for the crash adversary

We will resort to NP-completeness of the following problem, see [13].

Problem: GRAPH 3-COLORABILITY
Instance: A simple graph G.
Question: Can the nodes of G be colored with three colors in such a way that adjacent nodes have distinct colors?

We give a decision problem regarding broadcast protocols correct against the crash adversary that is NP-complete.

Problem: CRASH-ADVERSARY 3-RADIO BROADCAST
Instance: A radio network G.
Question: Is there a repeat-broadcast function for G that assigns at most three transmissions to each node such that the resulting broadcast protocol is correct against the crash adversary?

Theorem 9. *The problem* CRASH-ADVERSARY 3-RADIO BROADCAST *is* NP*-complete.*

Proof. We reduce the problem GRAPH 3-COLORABILITY to the problem considered. Let graph $F = (V, E)$ be an instance of GRAPH 3-COLORABILITY. Construct a directed radio network H as follows. There are three layers of nodes. The top layer L_0 consists of just the source s. The middle layer L_1 consists of all nodes $V = V[F]$ of the graph F. The last layer L_2 consists of the nodes that correspond to the edges in F, each node in L_2 representing one

edge in $E = E[F]$. The source s is an in-neighbor of each node in L_1. Nodes u and v in L_1 are in-neighbors of w in L_2 only if the edge $\langle u, v\rangle$ corresponds to node w. Observe that a node w in L_2 has in-degree equal to 2, while a node v in L_1 has its out-degree equal to its degree in F. Next we show the correctness of this reduction.

Let us assume that broadcasting in radio network H can be achieved with at most three transmissions per node by some protocol $\mathcal{B}$. Each node v in L_2 is vulnerable. By Lemma 5, one among the two neighbors u and v of w is a guard. Since there are only two of them, their repeat-broadcast values cannot be equal. Observe that none of these repeat-broadcast values is equal to 0, since otherwise the remaining neighbor of w could be crashed preventing node w from receiving the broadcast value. Define a coloring $\mathcal{C}$ of F by setting $\mathcal{C}(v) = \mathcal{B}(v)$, for a node v in $V[F] = L_1$. Since each node in L_2 corresponds to an edge in F, the protocol $\mathcal{B}$ yields a correct coloring of graph F by three colors.

Assume now that the nodes of graph F can be colored by 3 colors. Let $\mathcal{C}$ be such a coloring of F. We have $C(v) \neq C(u)$, for each edge $\{v, u\}$. Let the numbers 1, 2, and 3 be the names of colors. Define a broadcast protocol $\mathcal{B}$ for H by setting $\mathcal{B}(s) = 1$, for the source node s, then setting $\mathcal{B}(v) = \mathcal{C}(v)$, for each node v in L_1, and finally setting $\mathcal{B}(w) = 0$, for each node w in L_2. The repeat-broadcast value for each node is at most 3. The correctness of the protocol can be argued as follows. All the nodes in the middle layer L_1 receive the source message after a single transmission from the source. What remains to show is that any node w will eventually receive the message. There are two in-neighbors of w of distinct repeat-broadcast values. If exactly one of them is crashed, then the surviving node will relay the message to w. If none of them is crashed, then Lemma 1 applies. □

6.3. Hard for the node adversary

The problem to verify the correctness of a broadcast protocol against the edge adversary was shown to be solvable in polynomial time in Section 5.1. The corresponding problem for the node adversary turns out to be complete in co-NP, which we show in this section.

The reduction we give is from a problem about domino tilings. A *domino tile* is a unit square with sides parallel to the coordinate axes and with colors assigned to the sides. A tile is oriented, in the sense that its upper, right, lower, and left sides are fixed. A quadruple $\langle a, b, c, d\rangle$ of colors, possibly with repetitions, is a *domino type*. A domino tile *is of domino type* $\langle a, b, c, d\rangle$ when its upper, right, lower, and left sides are assigned the colors a, b, c, and d, respectively.

Tiles are placed in unit square regions of the plane determined by a grid of lines parallel to the coordinate axes and intersecting the axes at points of integer coordinates. We will consider square $n \times n$ regions of the grid, for some positive integer n. One among colors is distinguished, say, it is white. Let T be a set of domino types. A *T-tiling* of an $n \times n$ square in the plane is an assignment of tiles of the types in T to the unit squares in the $n \times n$ square subject to the following two constraints:

1. Any two adjacent tiles have matching colors on their common boundary.
2. The sides of tiles making the outer boundary are all colored white.

A domino tiling of $n \times n$ square grid can be interpreted as representing an execution of a non-deterministic Turing machine, where columns correspond to n contiguous memory cells and rows represent n consecutive configurations. Colors encode the tape alphabet, the position of the head and the state. The input is encoded by the bottom row of tiles, which could be assumed to be uniquely determined by the set of types and the number n.

Problem: SQUARE-IN-UNARY DOMINO TILING
Instance: A finite set T of domino tiles and an integer n in unary notation.
Question: Is there a T-tiling of the $n \times n$ square?

The interpretation of tilings, in which rows represent configuration, explains why SQUARE-IN-UNARY DOMINO TILING is an NP-complete problem, see [6]. We show that the tiling problem can be reduced to the complement of the following decision problem:

Problem: NODE-ADVERSARY CORRECTNESS RADIO BROADCAST
Instance: A network G with a distinguished source, and a repeat-broadcast function $\mathcal{B}$ for the nodes $V[G]$ of G.
Question: Is the broadcast protocol $\mathcal{B}$ correct against the node adversary?

Theorem 10. *The problem* NODE-ADVERSARY CORRECTNESS RADIO BROADCAST *is complete in* co-NP.

Proof. The complement of the problem belongs to NP because a strategy for the node adversary to make broadcast unsuccessful is a certificate that can be verified in polynomial time.

Given an instance (T, n) of SQUARE-IN-UNARY DOMINO TILING, we want to construct a radio network with the property that there is a T-tiling of the $n \times n$ square if an only if the node adversary has a strategy to prevent a successful broadcast in the network. The network we build has an $n \times n$ grid structure, in that some *boundary* nodes can be interpreted as located on boundaries between unit grid cells of the $n \times n$ grid while other *cell* nodes can be visualized as located inside the cells. Cell nodes are grouped in pairs; one of the cell nodes in a pair is the *left* node and the other one is called the *bottom* node of the pair.

Each boundary node represents a color and each pair of cell nodes represents a domino type. The following are restrictions on which colors and types are used for specific boundaries and cells. Boundary nodes for inner boundaries represent all the available colors of types in T. There is only one boundary node per each unit-cell outer boundary representing the white color. Pairs of cell nodes for inner cells represent all the available domino types in T, while pairs of cell nodes for cells on the boundary of the $n \times n$ square are exactly those that represent the types with the white color on the outer sides of the cell. The assignment of a color of the boundary is represented by exactly one node on the boundary receiving the source message. Such a representation results from a selection made by the adversary of the domino type for the cell.

The network is defined to include exactly the edges that make the following deliveries possible. Each node in a pair of cell nodes matching the colors on the left-hand-side and bottom boundaries receives the message. More precisely, the left node in the pair receives the message from the left-hand-side boundary node while the bottom node of the pair receives the message from the bottom boundary node. A boundary node on a vertical boundary with a cell to the right can transmit to exactly those left cell nodes in this cell that have the corresponding color on their left side. Similarly, a boundary node on a horizontal boundary with a cell above can transmit to exactly those bottom cell nodes in this cell that have the corresponding color on their bottom side.

In the first event of an execution, the transmission by the source delivers the message to all the boundary nodes on the outer boundary. In general, any following transmission, in an execution that does not result in a successful broadcast, can be interpreted as consistent with assigning domino types to some cells with the property that unique boundary nodes on the left-hand-side and bottom boundaries of these cells hold messages, until an $n \times n$ tiling has been determined. There is only one such a cell, at the bottom left corner, immediately after the first event in an execution.

There are two special nodes, called the *blocker* and the *finisher*, and a number of additional special nodes called *helpers*. These nodes have special roles to play, which is achieved by way of adding specific edge to the networks, as stipulated next. The finisher node can transmit directly to any other node in the network. The blocker node can transmit directly to the finisher node. If the finisher node receives the message, then it will perform sufficiently many transmissions to have one of them performed as the only node in the event. If the blocker node receives the message before the finisher node while no helper node has received the message yet, then the blocker has a sufficiently large repeat-broadcast value to perform sufficiently many transmissions to block the finisher node from ever receiving the message. Each helper node can transmit directly to the finisher node. When at least one helper receives the message, then also the finisher will receive the message eventually. The exact values of the repeat-broadcast function assigned to the nodes are explained later.

We want to have a network for which the only way for the adversary to inform the blocker node without informing the finisher node or any of the helper nodes is by way of simulating a domino tiling. The construction guarantees this property by the topology of the network. One component of the design provides that exactly one node on any boundary has to receive the message, in order to represent selecting a color for this boundary. Another component is that every left cell node in the top right corner cell can transmit to the blocker node, and no other node can do this. When such a node receives the message, but no helper node has received the message yet, then a tiling has been completed.

Next we describe the machinery to simulate a tile selection for a cell. Recall that a node v is said to transmit in an event when a message from v is delivered in the event. Consider a specific cell, and assume that both exactly one boundary node, say, x on the left-hand side and exactly one boundary node, say, y on the bottom side hold the message. We want these two nodes to transmit in the same event. Add a helper node for the cell with edges from any boundary node, like x and y, allowing these boundary nodes to transmit directly to the helper node. If a pair of boundary nodes transmit together in the same event, then there is a collision at the helper node. After the nodes x and y transmitted together, then exactly the nodes in pairs of cell nodes representing a tile with the color of x on the

left and the color of y at the bottom both receive the message. There may be many such pairs for the cell. Some pairs of cell nodes may have only one node holding the message while the other does not.

Consider a pair of cell nodes, say consisting of a and b, where a is a left node and b is a bottom one. Let node a can transmit directly to every boundary node on the top boundary, unless the outer boundary colored white is at the top. Similarly, let node b can transmit to every node on the right-hand side of the cell, unless the outer boundary colored white is on the right. To make only singleton nodes on these two boundaries receive the message, node a can transmit to every node on the right boundary representing a color different from the color of the type of the pair, while similarly node b can transmit to every node on the top boundary representing a color different from the color of the type of the pair; subject to the same restriction on white outer boundaries. We want the two nodes in only one such a pair transmit simultaneously. To enforce a simultaneous transmission of the two nodes in the same pair, add a special helper node for the pair, similarly as for x and y. Consider two such pairs. We want to have a mechanism that allows only one of them to transmit. To exclude separate transmissions, add a helper node for the pair of left components of the pair. Observe that a simultaneous transmission by two different pairs would result in no node on the top and right-hand side boundaries receiving the message, whichever do not belong to the outer boundary, so it is excluded. This property works except for the top right cell, which is a special case that is handled differently, as explained later.

Consider the top right cell. When a tiling has been successfully simulated, there is exactly one pair of cell nodes representing the tile with the left-hand side and bottom side colors as given by the boundary nodes holding messages, and with the right-hand and top sides colored white. Add edges that allow each cell node in this cell to transmit to the finisher node, while only one node, say left, of a pair can transmit to the blocker node. This guarantees that the two nodes in a pair need to receive the message, because two of them need to transmit simultaneously to inform the blocker while not informing the finisher.

Now we describe the values assigned by the repeat-broadcast function to the nodes of the network. The function has properties to provide the required functionality of special nodes, like finisher, blocker and helper ones. Each among the source, boundary and cell nodes is assigned 1 as the value of the repeat-broadcast function. Let r be the number of such nodes. The blocker nodes obtains r as its repeat-broadcast value. Name all the helper nodes as $b_0, b_1, \ldots, b_k$. Assign repeat-broadcast value $(r+1)2^i$ to node b_i, for $0 \leq i \leq k$. This assignment has the property that if any subset X of helpers receives the message and j is the largest index of a node b_j in this set, then there is an event in which b_j is the only node in X transmitting, because

$$\sum_{0 \leq \ell \leq j-1} (r+1)2^\ell = (r+1)(2^j - 1) < (r+1)2^j \ .$$

Finally, the finisher node receives the repeat-broadcast value larger by 1 than the sum of the repeat-broadcast values of all the remaining nodes.

The specifications of the network and of the repeat-broadcast function have the property that some among the nodes do not receive the message in an execution if and only if the blocker node receives the message while no helper node does, which in turn occurs if an only if an $n \times n$ tiling is represented by this execution. □

Problem EDGE-ADVERSARY WORK-BOUNDED RADIO BROADCAST was shown in Section 6.1 to be complete in NP. The corresponding problem for the node adversary is as follows.

Problem: NODE-ADVERSARY WORK-BOUNDED RADIO BROADCAST
Instance: A network G with a distinguished source, and a positive integer W.
Question: Is there a broadcast protocol for G correct against the node adversary and such that its work is at most W?

The problem NODE-ADVERSARY WORK-BOUNDED RADIO BROADCAST is in Σ_2^{P}. To see this, observe that a broadcast protocol is a certificate whose correctness is an instance of a problem in co-NP, since the correctness means that for every execution the broadcast is completed successfully. We conjecture that the problem is complete in Σ_2^{P}.

7. Conclusion

We have introduced models of asynchrony in radio networks and studied the problem of centralized broadcasting in these settings. The work of broadcast protocols for asynchronous networks needs sometimes to be exponential in the

size of the underlying networks. A reason for this phenomenon to occur is that the adversaries we consider represent the worst-case scenario of almost unlimited asynchrony. It would be interesting to define a weaker, but still natural, model of asynchrony in radio networks, for which polynomial-work protocols always exist.

The problems we considered concern centralized algorithms. Introducing a viable model of asynchronous ad hoc radio networks for which distributed communication protocols could be developed is an interesting topic of research to pursue.

Acknowledgement

The work of the second author was done while he was with the University of Colorado at Denver and Health Sciences Center.

References

[1] N. Alon, A. Bar-Noy, N. Linial, D. Peleg, A lower bound for radio broadcast, Journal of Computer and System Sciences 43 (1991) 290–298.
[2] H. Attiya, J. Welch, Distributed Computing: Fundamentals, Simulations, and Advanced Topics, 2nd ed., John Wiley, 2004.
[3] R. Bar-Yehuda, O. Goldreich, A. Itai, On the time complexity of broadcast in radio networks: An exponential gap between determinism and randomization, Journal of Computer and System Sciences 45 (1992) 104–126.
[4] I. Chlamtac, S. Kutten, On broadcasting in radio networks — Problem analysis and protocol design, IEEE Transactions on Communications 33 (1985) 1240–1246.
[5] I. Chlamtac, O. Weinstein, The wave expansion approach to broadcasting in multihop radio networks, IEEE Transactions on Communications 39 (1991) 426–433.
[6] B.S. Chlebus, Domino-tiling games, Journal of Computer and System Sciences 32 (1986) 374–392.
[7] B.S. Chlebus, L. Gąsieniec, A. Gibbons, A. Pelc, W. Rytter, Deterministic broadcasting in unknown radio networks, Distributed Computing 15 (2002) 27–38.
[8] B.S. Chlebus, L. Gąsieniec, A. Östlin, J.M. Robson, Deterministic radio broadcasting, in: Proc., 27th Colloquium on Automata, Languages and Programming, ICALP, in: LNCS, vol. 1853, 2000, pp. 717–728.
[9] A.E.F. Clementi, P. Crescenzi, A. Monti, P. Penna, R. Silvestri, On computing ad-hoc selective families, in: Proc., 5th International Workshop on Randomization and Approximation Techniques in Computer Science (APPROX-RANDOM), in: LNCS, vol. 2129, 2001, pp. 211–222.
[10] A.E.F. Clementi, A. Monti, R. Silvestri, Distributed broadcast in radio networks of unknown topology, Theoretical Computer Science 302 (2003) 337–364.
[11] A. Czumaj, W. Rytter, Broadcasting algorithms in radio networks with unknown topology, Journal of Algorithms 60 (2006) 115–143.
[12] I. Gaber, Y. Mansour, Broadcast in radio networks, Journal of Algorithms 46 (2003) 1–20.
[13] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[14] L. Gąsieniec, D. Peleg, Q. Xin, Faster communication in known topology radio networks, in Proc., 24th ACM Symposium on Principles of Distributed Computing, PODC, 2005, pp. 129–137.
[15] P. Indyk, Explicit constructions of selectors and related combinatorial structures, with applications, in Proc., 13th ACM-SIAM Symposium on Discrete Algorithms, SODA, 2002, pp. 697–704.
[16] D.R. Kowalski, A. Pelc, Broadcasting in undirected ad-hoc radio networks, Distributed Computing 18 (2005) 43–57.
[17] D.R. Kowalski, A. Pelc, Time of deterministic broadcasting in radio networks with local knowledge, SIAM Journal on Computing 33 (2004) 870–891.
[18] E. Kranakis, D. Krizanc, A. Pelc, Fault-tolerant broadcasting in radio networks, Journal of Algorithms 39 (2001) 47–67.
[19] E. Kushilevitz, Y. Mansour, An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks, SIAM Journal on Computing 27 (1998) 702–712.
[20] E. Kushilevitz, Y. Mansour, Computation in noisy radio networks, SIAM Journal on Discrete Mathematics 19 (2005) 96–108.
[21] N.A. Lynch, Distributed Algorithms, Morgan Kaufmann, 1996.
[22] G. Taubenfeld, Synchronization Algorithms and Concurrent Programming, Pearson/Prentice Hall, 2006.
[23] D.A. Wolfram, Solving generalized Fibonacci recurrences, The Fibonacci Quarterly 36 (1998) 129–145.